EJERCICIO 1: Arquitectura básica:

- Crear .py “arquitectura básica.py”
- Importar:
    - from dash import Dash, html
- Instanciar app:
    - App = Dash(__name__)
- Crear contenedor principal:
    - contenido = html.Div(“Hola Dash!)
- Layout básico:
    - App.layout = contenido
- Callbacks explicar pero dejar comentado
- Ejecución de la app
    - If __name__ == “__main__”:

```
App.run_server(debug = True)
```

EJERCICIO 2: Conexión a BBDD:

- Crear .py “functions.py”
- Importar:
  - Import pyodbc
  - Import pandas as pd
- Crear función de conexión:
  - def consulta(querystr):

```
cnxn = pyodbc.connect(
        “Driver= {SQL Server};”
        “Server= 172.19.10.226,1533;”
        “Database=SGP_SIPE;”
        “Trusted_Connection=no;”
        “UID=user_cuadrosM;”
        “PWD=w$gh84Con2”
        )
result = pd.read_sql_query(querystr, con = cnxn)
cnxn.close()
return result
```

- Comprobar conexión: crear notebook “prueba_conexion.ipynb”
  - import functions as f
  - data = f.consulta(“SELECT * FROM SGP_CUADROSMANDO.cm.resumen_flota)
  - data.head()

EJERCICIO 3: Primer Dashboard:

Dashboard sin interactividad, 1 fila, 3 columnas:

Grafico barras desembarques por censo, tabla desembarques comunidad, mapa burbujas desembarques por puerto

- crear carpeta ejer_3
- Traer archivo “functions”
- Crear carpeta “moduls” con su “ ”__init__.py”
- Importaciones:
    - From dash import Dash, html, dcc
    - EXPLICAR FORMAS DE DAR FORMATO (html, css bootstrap)
    - Import dash_bootstrap_componets as dbc

- Instanciar la app (explicar external_stylesheets y meta_tags)
    - App = Dash(__name__, external_stylesheets= [dbc.themes.BOOTSTRAP], meta_tags = [{“name”: “viewport”, “content”: “width, initial-scale=1”}])

- CREAR ESQUEMA DE ESTRUCTURA SIN ESTILOS
- Título:
    - titulo = html.H1(“Primer Dashboard con Dash”)
- Resumen:
    - Resumen = html.P(“Estamos aprendiendo a usar Dash para crear dashboards interactivos”)
- Contenido:
    - Html.Div([

```
Dbc.Row([
        dbc.Col(html.P(“grafico capturas censo”), width = 4),
        dbc.Col(html.P(“tabla CCAAs”), width= 4),
        dbc.Col(html.P(“mapa”), width = 4)
        ])
])
```

- Layout
    - app.layout = dcc.Loading(type= “circle”, fullscren= True, children = [titulo, resumen, contenido])

```
if __name__ == “__main__”:
    app.run_server(debug= True)
```

EJERCICIO 3.1 GRÁFICO CAPTURAS POR CENSO

Usamos plotly por su interactividad

- Dentro de moduls crear “graf_censo.py)
- Designar el path a la carpeta raíz y importaciones
    - Import sys
    - Import os
    - sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), “..”)))
    - from dash import dcc
    - import plotly.express as px
    - import pandas as pd
    - import functions as f
- crear función de gráfico

```
def graf_censo():
        data = f.consulta(“SELECT * FROM
        SGP_CUADROSMANDO.cm.ccaa_desembarques)

        data = data[data[“año”] == 2024]
        data = data.groupby(“Censo”)[“Peso”].sum().reset_index()
        data = data.sort_values(by = “Peso”, ascending = False).nlargest(6,
        “Peso”)

        bar = px.bar(data, x= “Censo”, y= “Peso”)

        fig_bar = dcc.Graph(id= “graf_censo”, figure= bar)
        return figure
```

- Ejecutar y ver resultado, después añadir estilo al gráfico

```
Bar.update_layout(
        Xaxis_title= “”,
        Xaxis_tickangle = 30,
        Height = 500,
        Margin = dict(l= 10, r= 20, t= 0, b= 0)
        )
```

EJERCICIO 3.2 TABLA CAPTURAS POR CCAA

- Dentro de moduls crear “tabla_desembarques.py”
- Importaciones:
    - Import sys
    - Import os
    - sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), “..”)))

    - from dash import dash_table
    - import pandas as pd
    - import functions as f
- Crear función de tabla SIN ESTILO
  def tabla_desembarques():
    data = f.consulta("SELECT * FROM SGP_CUADROSMANDO.cm.ccaa_desembarques WHERE año = 2024")

    data = data.groupby("CCAADesembarque")["Peso"].sum().reset_index()

    tabla = dash_table.DataTable(id = "tabla-desembarques",

      columns= [{"name": col, "id": col} for col in data.columns],

      data = data.to_dict("records")
)

    return tabla

- AÑADIR AL MAIN Y EJECUTAR
- AÑADIR ESTILOS A LA TABLA

```
style_header={"fontWeight": "bold", "backgroundColor": "#ebf5fb",
"fontFamily": "Arial", "minWidth": 350, "whiteSpace": "normal", "text-
align": "center", "fontSize": "15px"},
style_data={"backgroundColor": "#ffffff", "color": "#333333",
"fontFamily": "Arial", "fontSize": "13px"},
style_table={"overflowY": "auto"},
style_cell={"minWidth": 100, "width": "auto", "text-align": "left"},
style_data_conditional=[
            {"if": {"row_index": "odd"}, "backgroundColor": "#f8f9f9"},
            {"if": {"row_index": "even"}, "backgroundColor": "#ffffff"}
                                ],
cell_selectable=False,
fixed_rows={"headers": True},
sort_action="native",
filter_action= "native",
page_size=12
             )
```

EJERCICIO 3.3 MAPA CAPTURAS POR PUERTO

- Dentro de moduls crear "mapa_desembarques.py"
- Importaciones:
    - Import sys
    - Import os

- sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), "..")))
- from dash import html
- import pandas as pd
- import folium
- from shapely import wkt
- from folium.plugins import Fullscreen
- import functions as f

- Crear función
  - Data puertos

```
def mapa_desembarques():
    sqlstr = f"""
        Select prt.Descripcion, prt.Coordenadas.STAsText() as Coordenadas
from
        (select bi.IdPuertoBase IdPuerto from [censo].[BuqueEstado] be
inner join [censo].[BuqueIdentificacion] bi on be.IdBuque=bi.idbuque
        inner join fenix.Puerto prt on bi.IdPuertoBase=prt.id
        inner join cat.Provincia pr on pr.id=prt.IdProvincia
        where  GETDATE() between be.FcEfectoInicial and be.FcEfectoFinal
and
            GETDATE() between bi.FcEfectoInicial and bi.FcEfectoFinal and
            idtipoestado in (1,5)
        group  by bi.IdPuertoBase) A
        inner join fenix.Puerto prt on A.IdPuerto=prt.id
        """

    data = f.consulta(sqlstr)
    data = data.rename(columns= {"Descripcion": "PuertoDesembarque"})
    data = data.dropna()
```

  - Data buques

```
    data_buques = f.consulta("SELECT * FROM
SGP_CUADROSMANDO.cm.ccaa_desembarques")
    data_buques = data_buques[data_buques["Año"] == 2024]
    data_buques =
data_buques.groupby("PuertoDesembarque")["Peso"].sum().reset_index()

 data_total = pd.merge(data, data_buques, how = "left", on =
"PuertoDesembarque")
```

  - Calcular centro y crear mapa

```
    # Calcular centro
```

```
    data["geometry"] = data["Coordenadas"].apply(wkt.loads)
    data["lat"] = data["geometry"].apply(lambda x: x.y)
    data["lon"] = data["geometry"].apply(lambda x: x.x)
    center_lat = data["lat"].mean()
    center_lon = data["lon"].mean()

    mapa = folium.Map(location=[center_lat, center_lon], zoom_start=4,
height= 450)
```

- Añadir marcadores al mapa

```
# Añadir marcadores al mapa
    for _, row in data_total.iterrows():
        # Convertir la coordenada WKT a un objeto de Shapely
        coordenada = wkt.loads(row["Coordenadas"])

        # Obtener latitud y longitud
        lat, lon = coordenada.y, coordenada.x

        # Añadir el marcador en el mapa
        folium.CircleMarker(
            location=[lat, lon],
            radius=5 + row["Peso"] / 1000 * 0.001,
            color="#2e86de",
            fill=True,
            fill_color="#3498db",
            fill_opacity=0.6,
            popup=f'{row["PuertoDesembarque"]}<br>Peso desembarque:
{row["Peso"] / 1000} T'
        ).add_to(mapa)

    # Mostrar el mapa
    Fullscreen(position="topright").add_to(mapa)
    map_html_content = mapa.get_root().render()
```

```
    return dbc.Card(
        dbc.CardBody(
            html.Iframe(
                srcDoc=map_html_content,
                width="100%",
                height=450,
                style={"border": "none", "height": "100%"}
            ),
```

```
            style={"height": "450px", "padding": "0px", "width": "100%",
"display": "flex", "justify-content": "center", "align-items":
"center"}
        ),
        style={"padding": "0px", "border": "none", "height": "100%",
"display": "flex", "justify-content": "center", "align-items":
"center"}
    )
```

EJERCICIO 3.4 DAR ESTILO AL DASHBOARD

```
contenido = html.Div([
    dbc.Row([
        dbc.Col(dbc.Card([
            dbc.CardHeader(html.H4("Capturas por Censo", style= {"text-
align": "center"}), style= {"backgroundColor": "#f9feff"}),
            dbc.CardBody(gc.graf_censo())
            ]), width= 4),

        dbc.Col(dbc.Card([
            dbc.CardHeader(html.H4("Desembarques por CCAA", style=
{"text-align": "center"}), style= {"backgroundColor": "#f9feff"}),
            dbc.CardBody(tb.tabla_desembarques())
            ]), width= 4),

        dbc.Col(dbc.Card([
            dbc.CardHeader(html.H4("Desembarques por Puerto", style=
{"text-align": "center"}), style= {"backgroundColor": "#f9feff"}),
            dbc.CardBody(md.mapa_desembarques())
            ]), width= 4),
    ])
], style= {"padding": "20px"})
```